

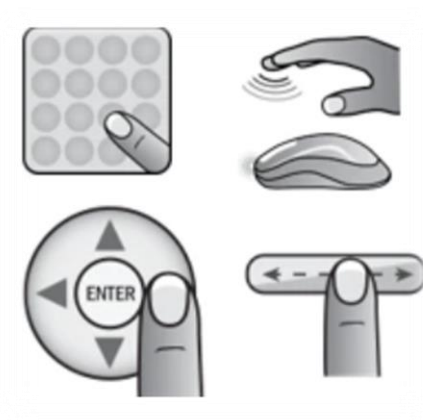
# Introduction to CapSense in ModusToolbox

Bragadeesh Viswanathan  
Applications Engineer  
21st May, 2020



# Agenda

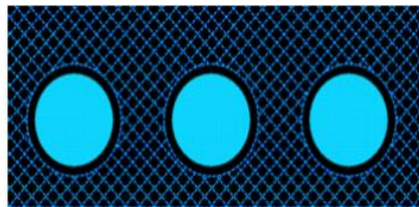
- › Introduction to CapSense technology
- › Self-Capacitance vs Mutual Capacitive sensing
- › High Level Hardware Architecture
- › Basic terminologies
- › CapSense Tuning
- › CapSense Middleware library
- › CapSense Configurator in ModusToolbox 2.x
- › Demo Project
- › CapSense Data Structure
- › CapSense Tuner GUI
- › Advanced Topics



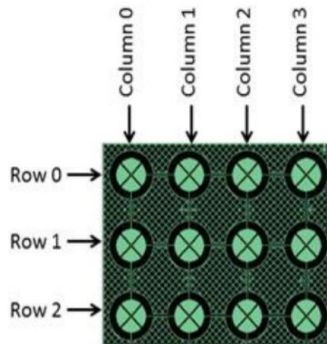
# CapSense Technology

- › CapSense technology measures changes in capacitance between a plate (the sensor) and its environment to detect the presence of a finger on or near a touch surface
- › Features:
  - Robust touch sensing
  - Liquid tolerant operation
  - Low electromagnetic interference (EMI) performance
  - SmartSense™ Auto-Tuning technology
  - Low power consumption
  - Wide operating voltage range (1.71 V – 3.6 V) – PSoC 6
  - Proximity sensing

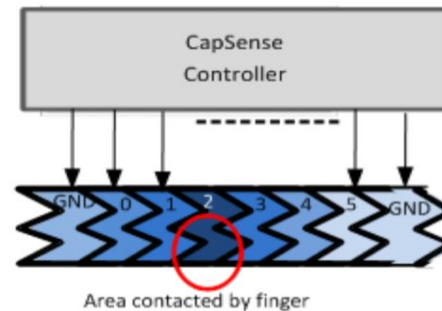
# CapSense Sensors



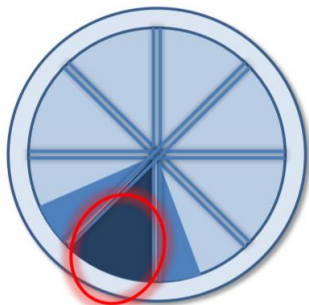
Buttons



Matrix Buttons

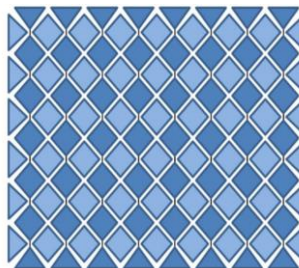


Linear Slider

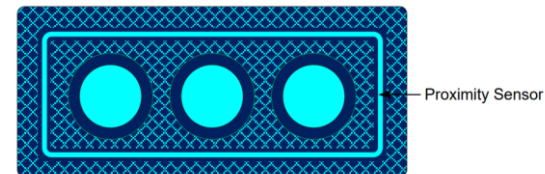


Area contacted by finger

Radial Slider



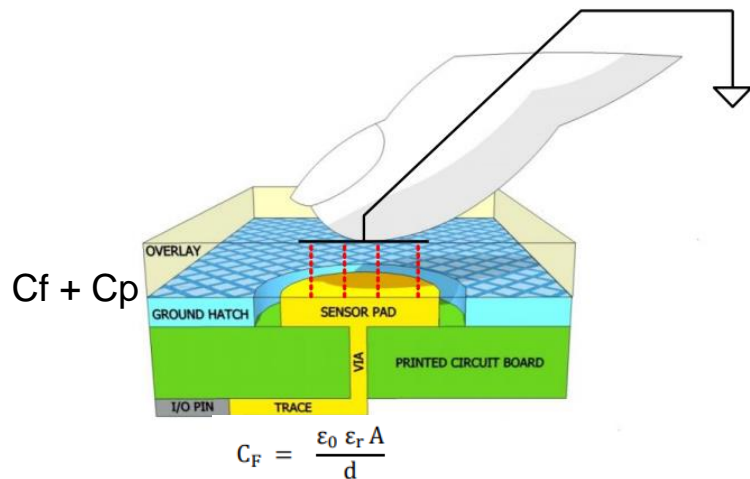
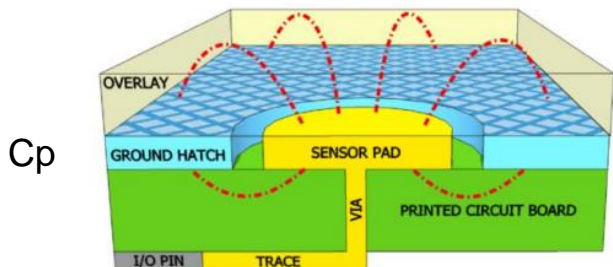
Touchpad/Trackpad



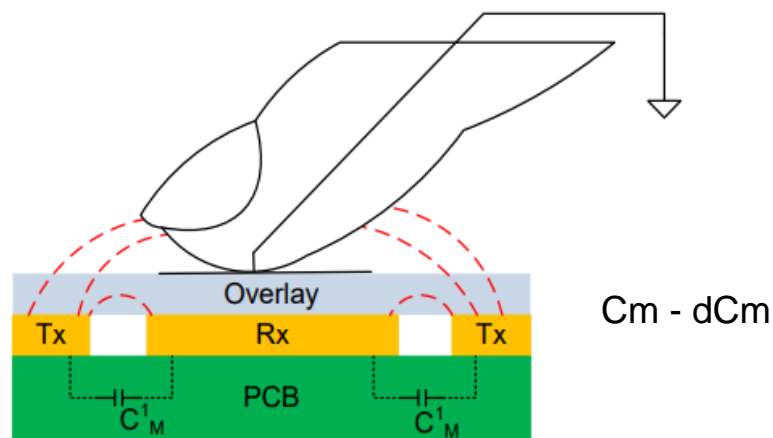
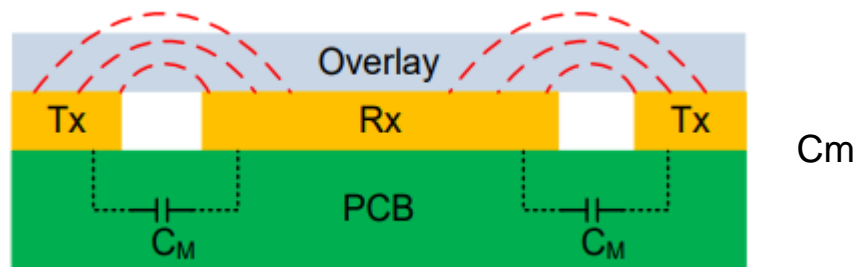
Proximity sensor

# Self Capacitance vs Mutual Capacitance

## Self Capacitance



## Mutual - Capacitance

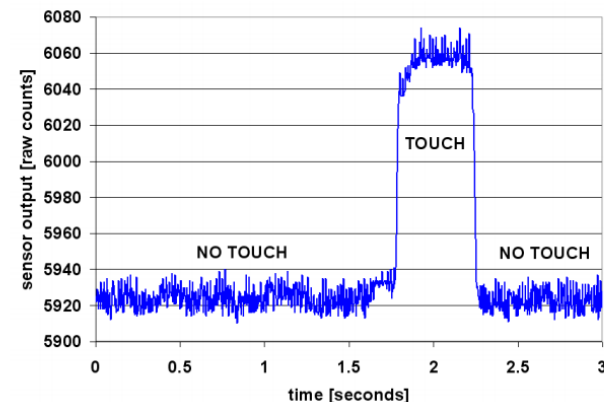
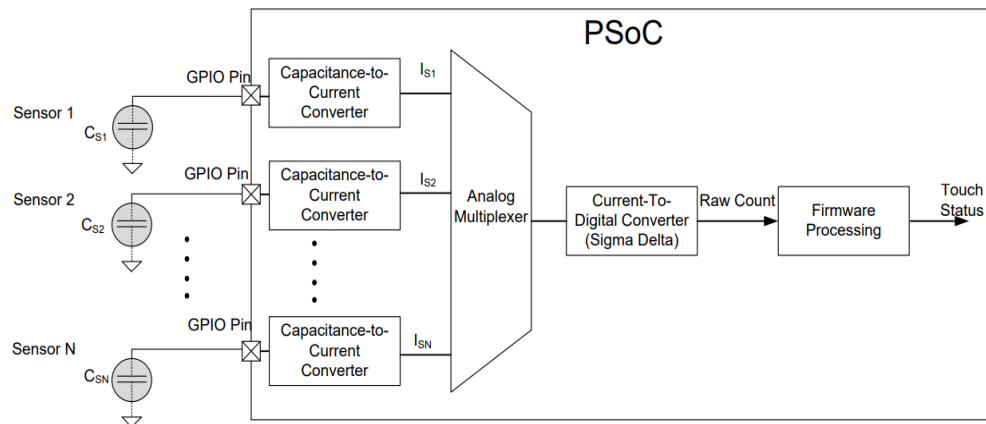


## When to use CSD , When to use CSX

Parameter/ Requirement	CSD	CSX
Pin constraints	Requires more pins	Requires Less pins <sup>(1)</sup>
High Cp designs	✗	✓
Proximity sensing	✓	✗
Overlay thickness	Thick overlays	Thin overlays (1 - 4 mm)
Liquid tolerance	✓	✓ <sup>(2)</sup>
Multi touch performance in matrix buttons	✗ <sup>(3)</sup>	✓
Spring sensors	✓	✗
VDD noise	✓ <sup>(4)</sup>	✗

- > (1) If sensors required is greater than 7. If there are X \* Y sensors requirement, CSX requires only X + Y GPIO whereas CSD requires X \* Y GPIO
- > (2) CSX + CSD for liquid tolerance. It is a special firmware. Contact our community support for further information.
- > (3) Ghost touch – (wrong reporting of fingers upon multi-touch)
- > (4) IDAC sourcing mode

# CapSense block - CSD



- › Each GPIO has a switched-capacitance circuit that converts the sensor capacitance into an equivalent current
- › An analog multiplexer then selects one of the currents and feeds it into the current to digital converter
- › The current to digital converter is similar to a sigma delta ADC. The output count of the current to digital converter, known as raw count, is a digital value that is proportional to the self-capacitance between the electrodes.

## CapSense – Basic terminologies

### › RawCounts

- Digital count value that is proportional to the self-capacitance between the electrodes in case of CSD and Mutual capacitance between electrodes in case of CSX

### › Baseline

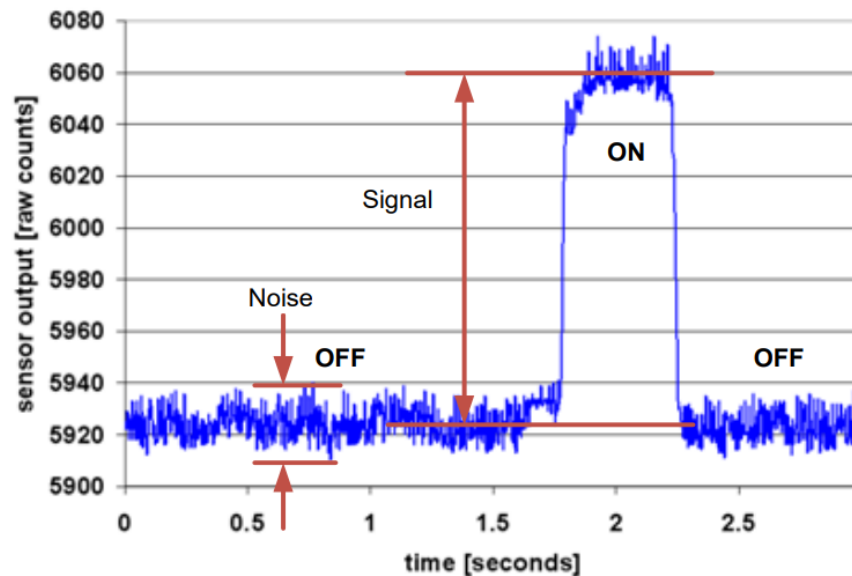
- Filtered rawcounts at no touch condition

### › Difference Counts

- $\text{RawCounts}(\text{with touch}) - \text{Baseline} (\text{Signal})$

### › Signal to Noise ratio

- Ratio of CapSense signal to CapSense noise. Recommended SNR is 5 : 1





# CapSense Tuning

---

Why should you tune sensors?

- › Reliable touch detection
- › Power consumption
- › Response time
- › Achieving Noise Immunity
- › Less Emissions

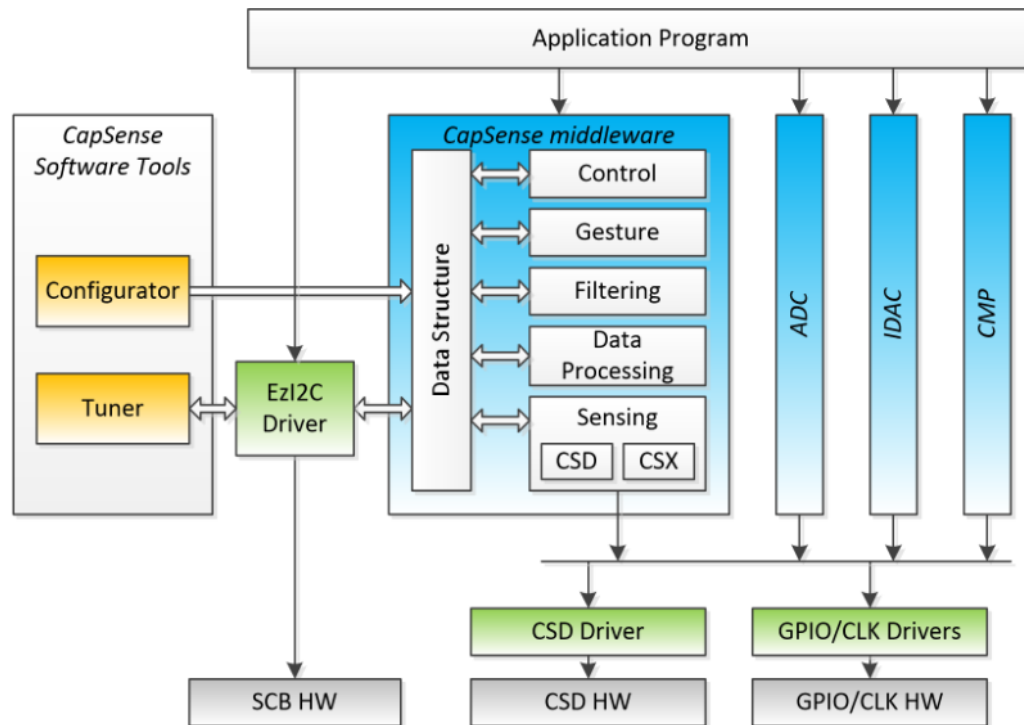
## CapSense Tuning – SmartSense vs Manual Tuning

---

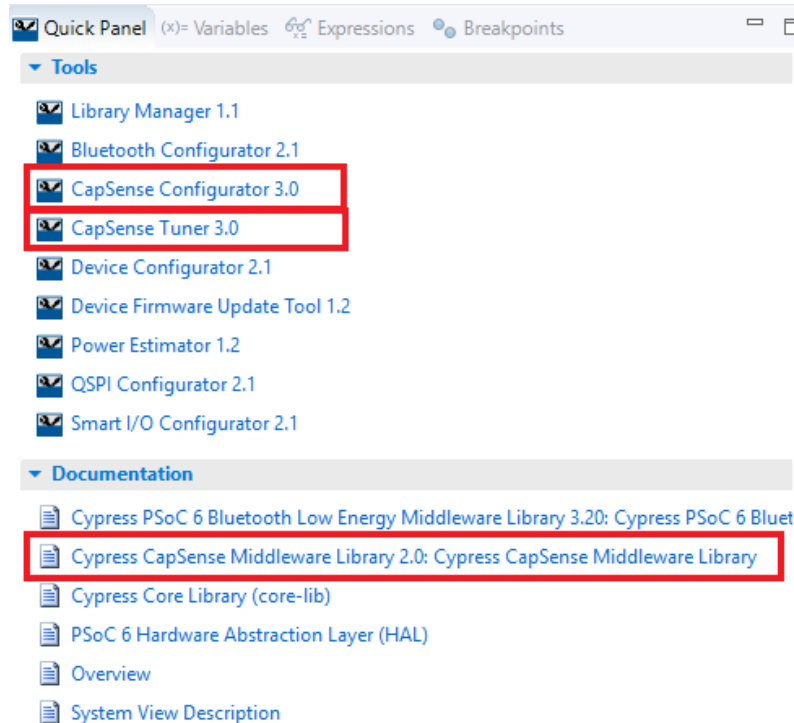
- › SmartSense Autotuning:
  - All parameters – set by component
  - Reduced design cycle time and easy to use
  - Limitations on memory and supported parasitic capacitance
  - Less control over performance
- › Manual Tuning:
  - Strict control over parameter settings such as response time, power consumption and noise immunity.
  - Higher parasitic capacitance designs

# CapSense Middleware Library

- › The CapSense Configurator tool, which is a configuration wizard to create and configure CapSense widgets
- › API to control the design from the application program.
- › The CapSense Tuner tool for real-time tuning, testing, and debugging, for easy and smooth designing of human interfaces on customer products



# CapSense Configurator and CapSense Tuner GUI



↑ This PC > Windows (C:) > Users > brvi > ModusToolbox > tools\_2.1 > capsense-configurator

Name	Date modified	Type	Size
bearer	07-Apr-20 3:33 PM	File folder	
docs	07-Apr-20 3:33 PM	File folder	
firmware	07-Apr-20 3:33 PM	File folder	
iconengines	07-Apr-20 3:33 PM	File folder	
imageformats	07-Apr-20 3:33 PM	File folder	
platforms	07-Apr-20 3:33 PM	File folder	
printsupport	07-Apr-20 3:33 PM	File folder	
styles	07-Apr-20 3:33 PM	File folder	
<b>capsense-configurator.exe</b>	25-Mar-20 4:35 AM	Application	14,088 KB
capsense-configurator.png	25-Mar-20 4:32 AM	PNG File	1 KB
capsense-configurator-cli.exe	25-Mar-20 4:35 AM	Application	13,476 KB
<b>capsense-tuner.exe</b>	25-Mar-20 4:36 AM	Application	17,085 KB
capsense-tuner.png	25-Mar-20 4:32 AM	PNG File	1 KB
configurator.xml	25-Mar-20 4:32 AM	XML Source File	1 KB
CyBridge.dll	25-Mar-20 4:36 AM	Application extens...	3,504 KB
libusb-1.0.dll	25-Mar-20 4:36 AM	Application extens...	146 KB
Qt5Core.dll	25-Mar-20 4:36 AM	Application extens...	5,986 KB
Qt5Gui.dll	28-Jan-19 10:45 PM	Application extens...	6,308 KB
Qt5Network.dll	28-Jan-19 10:45 PM	Application extens...	1,285 KB
Qt5PrintSupport.dll	28-Jan-19 10:50 PM	Application extens...	311 KB
Qt5Svg.dll	29-Jan-19 8:51 AM	Application extens...	323 KB
Qt5Widgets.dll	28-Jan-19 10:50 PM	Application extens...	5,446 KB
Qt5XmlPatterns.dll	29-Jan-19 9:26 AM	Application extens...	3,240 KB
version.xml	25-Mar-20 4:32 AM	XML Source File	1 KB

# CapSense Configurator

C:/Users/brvi/MTB\_2\_1\_TEST/CapSense\_Tuning\_over\_BLE\_Server/libs/TARGET\_CY8CKIT...

File Edit View Help

Basic Advanced Pins

Move up Move down Delete CSD tuning mode: Manual tuning

Type	Name	Sensing Mode	Sensing Element(s)				Finger Capacitance
○	Button0	CSX (Mutual-cap)	1	Rx	1	Tx	N/A
○	Button1	CSX (Mutual-cap)	1	Rx	1	Tx	N/A
»	LinearSli...	CSD (Self-cap)	5	Segments			N/A
+							

Sensor resources

CSD electrodes: 5 CSX electrodes: 4 Pins required: 11

Notice List

Fix	Description	Location
-----	-------------	----------

Ready Device: PSoC 6

- › Used to create and configure CapSense widgets
- › Generate code to control application firmware

## Demo Project – CapSense Initialization

- › Include `cycfg_capsense.h` to include CapSense Middleware and Configurator settings into your project
- › Initialize CapSense
  - CapSense interrupt configuration
  - Initialize CapSense hardware
  - Initialize CapSense interrupt
  - Enable CapSense interrupt
  - Initialize CapSense firmware modules

```

/* CapSense interrupt configuration */
const cy_stc_sysint_t CapSense_interrupt_config =
{
    .intrSrc = CYBSP_CSD_IRQ,
    .intrPriority = CAPSENSE_INTR_PRIORITY,
};

/* Capture the CSD HW block and initialize it to the default state. */
status = Cy_CapSense_Init(&cy_capsense_context);

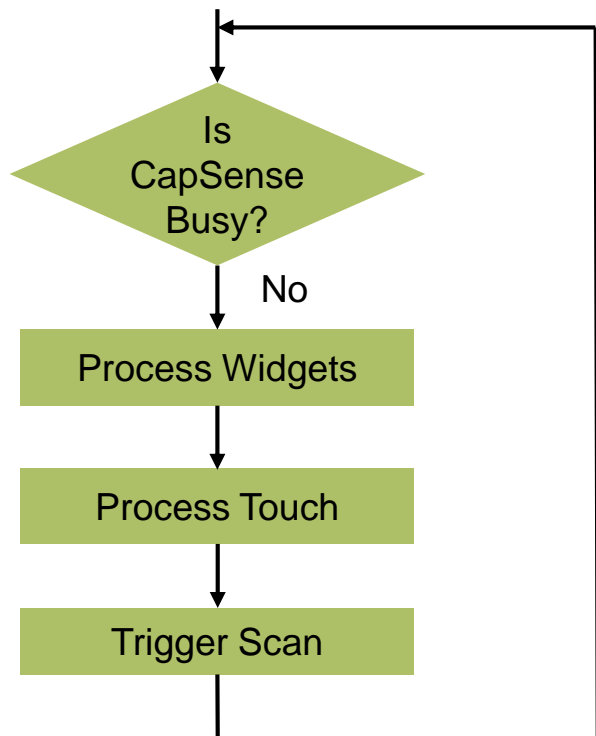
if(CYRET_SUCCESS == status)
{
    /* Initialize CapSense interrupt */
    Cy_SysInt_Init(&CapSense_interrupt_config, capsense_isr);
    NVIC_ClearPendingIRQ(CapSense_interrupt_config.intrSrc);
    NVIC_EnableIRQ(CapSense_interrupt_config.intrSrc);

    /* Initialize the CapSense firmware modules. */
    status = Cy_CapSense_Enable(&cy_capsense_context);
}

```

# Demo Project – CapSense Scanning and Processing results

## › Application for ever loop



```

/* Start first scan */
Cy_CapSense_ScanAllWidgets(&cy_capsense_context);

for(;;)
{
    if(CY_CAPSENSE_NOT_BUSY == Cy_CapSense_IsBusy(&cy_capsense_context))
    {
        /* Process all widgets */
        Cy_CapSense_ProcessAllWidgets(&cy_capsense_context);

        /* Process touch input */
        process_touch();

        /* Start next scan */
        Cy_CapSense_ScanAllWidgets(&cy_capsense_context);
    }
}
  
```

## Demo Project – Check Widget Status

- › **Cy\_CapSense\_IsSensorActive()**
  - Reports whether the specified sensor in the widget detected touch.
  
- › **Cy\_CapSense\_GetTouchInfo()**
  - Reports the details of touch position detected on the specified touchpad, matrix buttons or slider widgets.
  
- › **Similar APIs**
  - Cy\_CapSense\_IsWidgetActive()
  - Cy\_CapSense\_IsProximitySensorActive()
  - Cy\_CapSense\_IsAnyWidgetActive()

```

/* Get button 0 status */
button0_status = Cy_CapSense_IsSensorActive(
    CY_CAPSENSE_BUTTON0_WDGT_ID,
    CY_CAPSENSE_BUTTON0_SNS0_ID,
    &cy_capsense_context);

/* Get button 1 status */
button1_status = Cy_CapSense_IsSensorActive(
    CY_CAPSENSE_BUTTON1_WDGT_ID,
    CY_CAPSENSE_BUTTON1_SNS0_ID,
    &cy_capsense_context);

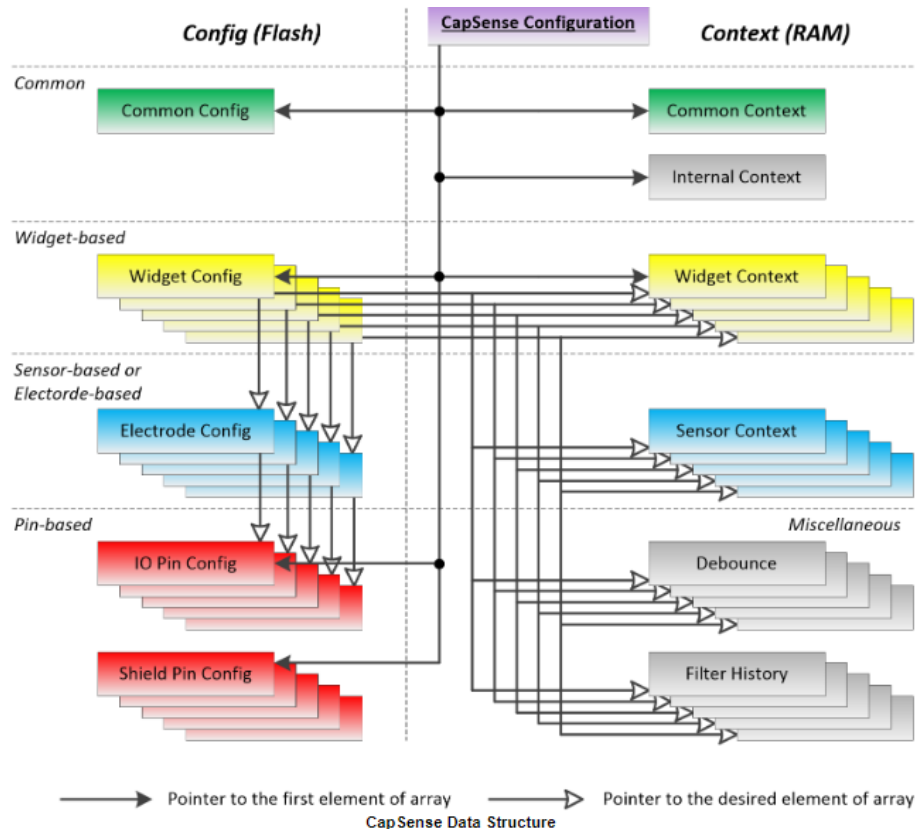
/* Get slider status */
slider_touch_info = Cy_CapSense_GetTouchInfo(
    CY_CAPSENSE_LINEARSLIDER0_WDGT_ID, &cy_capsense_context);
slider_touch_status = slider_touch_info->numPosition;
slider_pos = slider_touch_info->ptrPosition->x;

```



# CapSense Data Structure

- The Data Structure is the only data container in the CapSense middleware.
- It serves as storage for the configuration and the output data.
- All CapSense modules use the data structure for the communication and data exchange.
- Data is split between RAM and Flash to achieve a reasonable balance between resources consumption and configuration / tuning flexibility at runtime and compile time



## Demo Project – Adding Tuner Support using EZI2C interface

- › Initialize an EZI2C interface to communicate with CapSense Tuner GUI
- › Expose cy\_capsense\_tuner structure as the read/ write buffer in EZI2C interface

```
/* Populate slave configuration structure slave_config1 */
cyhal_ezi2c_slave_cfg_t slave_config1 =
{
    .slave_address = SLAVE_ADDRESS,
    .buf = (uint8_t *)&cy_capsense_tuner,\
    .buf_size = sizeof(cy_capsense_tuner),
    .buf_rw_boundary = sizeof(cy_capsense_tuner)
};

/* Populate the EZI2C configuration structure */
cyhal_ezi2c_cfg_t ezi2c_config =
{
    .two_addresses = false,
    .enable_wake_from_sleep = true,
    .data_rate = CYHAL_EZI2C_DATA_RATE_400KHZ,
    .slave1_cfg = slave_config1,
    .sub_address_size = CYHAL_EZI2C_SUB_ADDR16_BITS
};

/* Initialize EZI2C */
cyhal_ezi2c_init(&ezi2c_obj, P6_1, P6_0, NULL, &ezi2c_config);
```

## Demo Project – Adding Tuner Support using EZI2C interface

- › Calling Cy\_CapSense\_RunTuner() from the application is optional.

```
for(;;)
{
    if(CY_CAPSENSE_NOT_BUSY == Cy_CapSense_IsBusy(&cy_capsense_context))
    {
        /* Process all widgets */
        Cy_CapSense_ProcessAllWidgets(&cy_capsense_context);

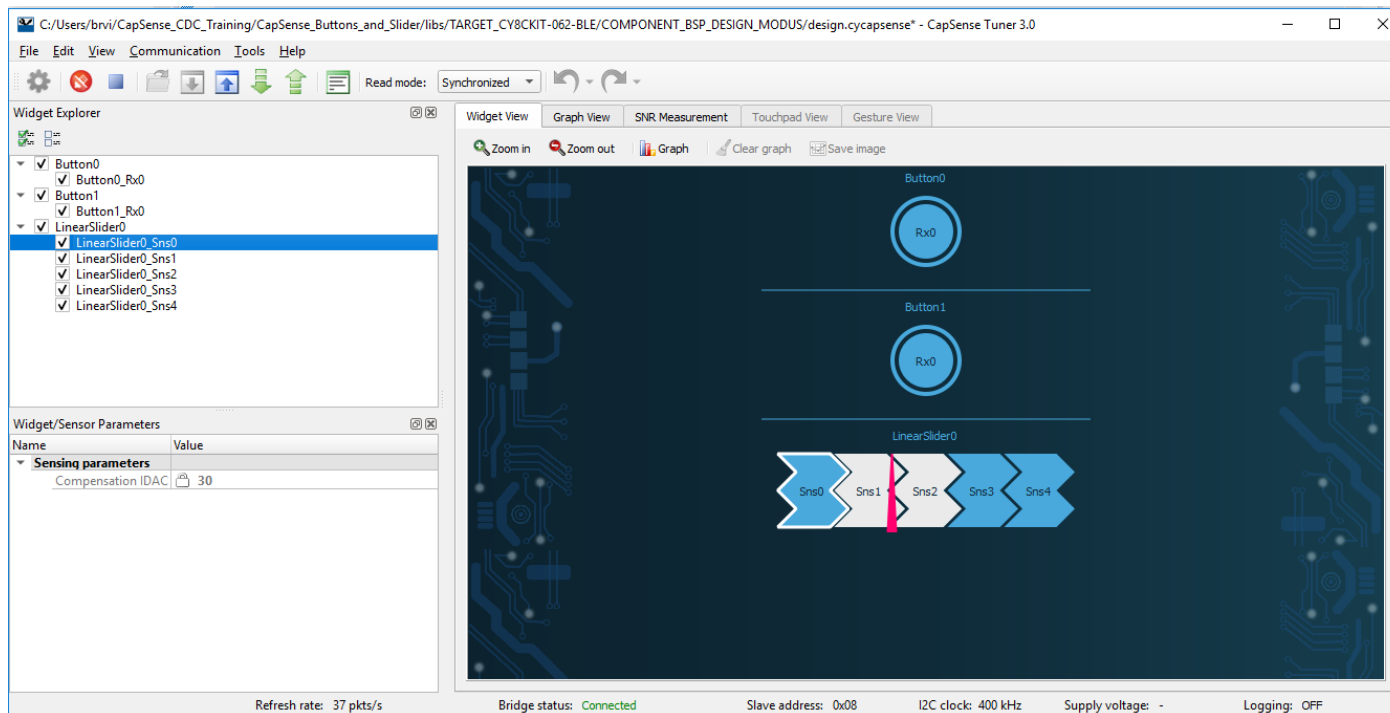
        /* Process touch input */
        process_touch();

        /* Establishes synchronized operation between the CapSense
         * middle ware and the CapSense Tuner tool.
         */
        Cy_CapSense_RunTuner(&cy_capsense_context);

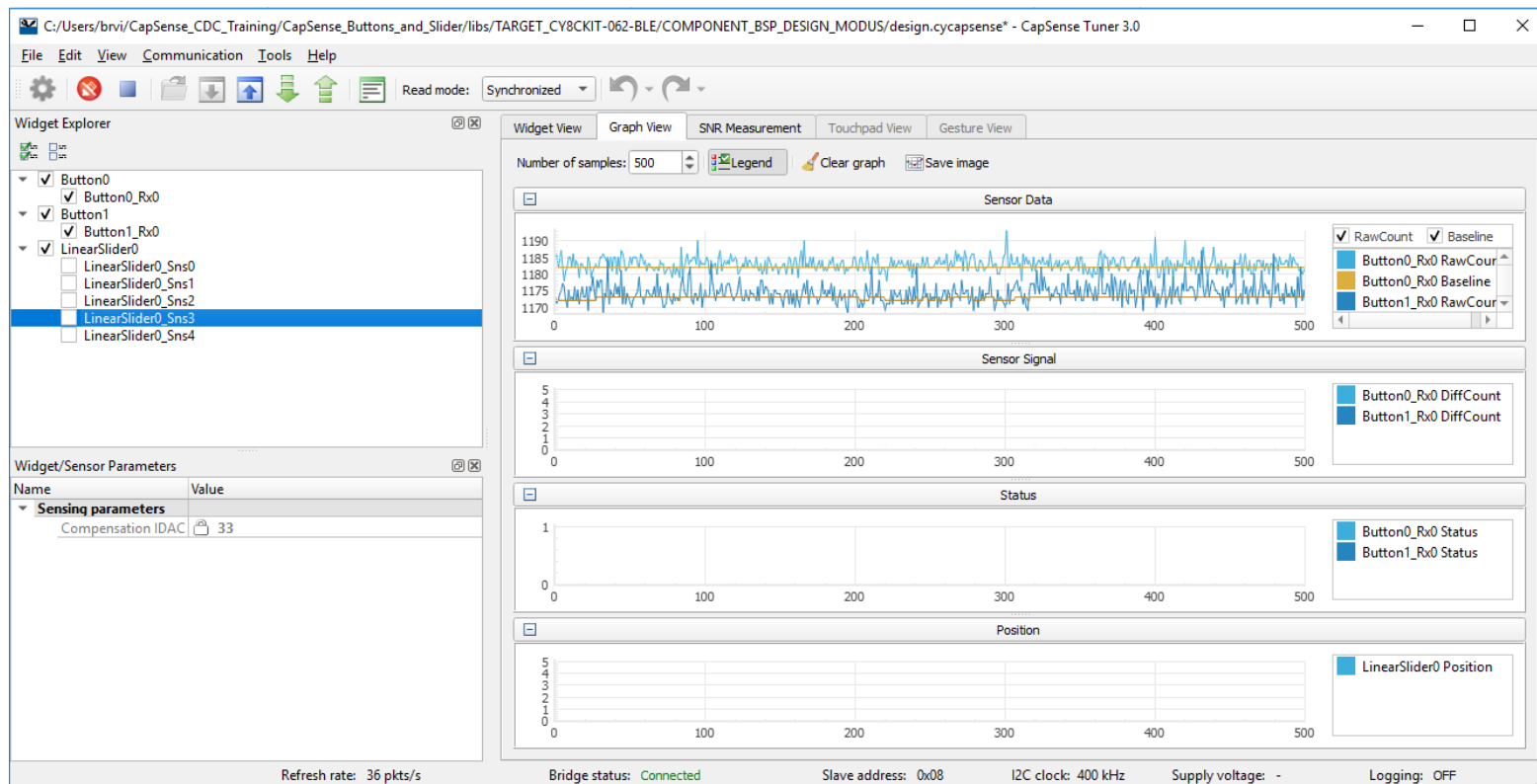
        /* Start next scan */
        Cy_CapSense_ScanAllWidgets(&cy_capsense_context);
    }
}
```

# CapSense Tuner GUI

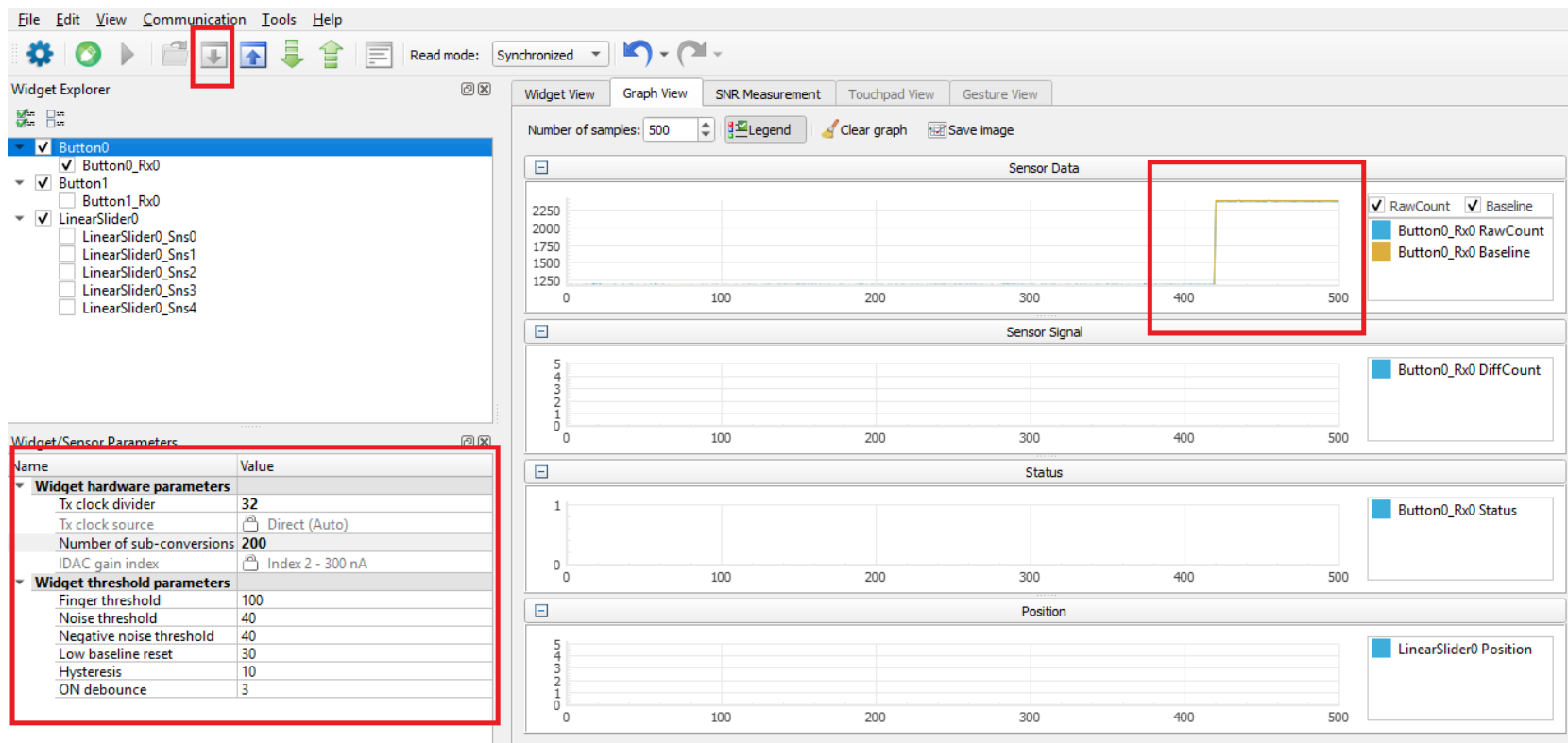
- › The CapSense Tuner tool for real-time tuning, testing, and debugging, for easy and smooth designing of human interfaces on customer products



# CapSense Tuner GUI – View CapSense raw data



# CapSense Tuner GUI – Change hardware/ software parameters



The screenshot displays the CapSense Tuner GUI interface. The top menu bar includes File, Edit, View, Communication, Tools, and Help. The toolbar contains icons for settings, execution, and data management. The 'Widget Explorer' on the left lists various widgets, with 'Button0' and its sub-components selected. The 'Widget/Sensor Parameters' table at the bottom left is highlighted with a red box, showing hardware and threshold parameters. The main area features four graphs: 'Sensor Data', 'Sensor Signal', 'Status', and 'Position'. The 'Sensor Data' graph is highlighted with a red box, showing a step function for 'RawCount' and 'Baseline'. The 'Read mode' is set to 'Synchronized'.

**Widget/Sensor Parameters**

Name	Value
<b>Widget hardware parameters</b>	
Tx clock divider	32
Tx clock source	Direct (Auto)
Number of sub-conversions	200
IDAC gain index	Index 2 - 300 nA
<b>Widget threshold parameters</b>	
Finger threshold	100
Noise threshold	40
Negative noise threshold	40
Low baseline reset	30
Hysteresis	10
ON debounce	3

**Sensor Data Graph**

Number of samples: 500

Legend: RawCount (blue), Baseline (yellow)

Button0\_Rx0 RawCount (Blue line)

Button0\_Rx0 Baseline (Yellow line)

**Sensor Signal Graph**

Button0\_Rx0 DiffCount (Blue line)

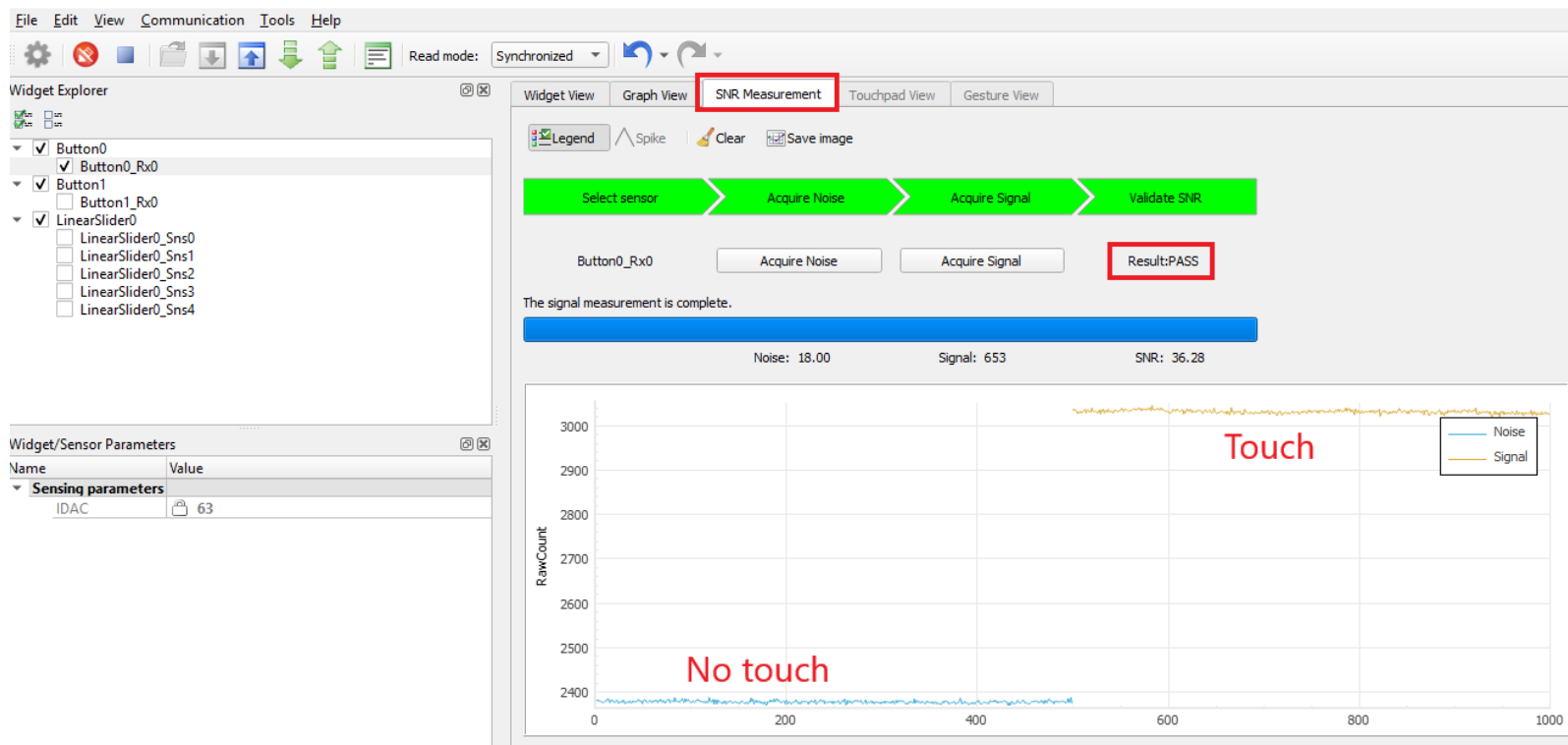
**Status Graph**

Button0\_Rx0 Status (Blue line)

**Position Graph**

LinearSlider0 Position (Blue line)

# CapSense Tuner GUI – SNR Performance

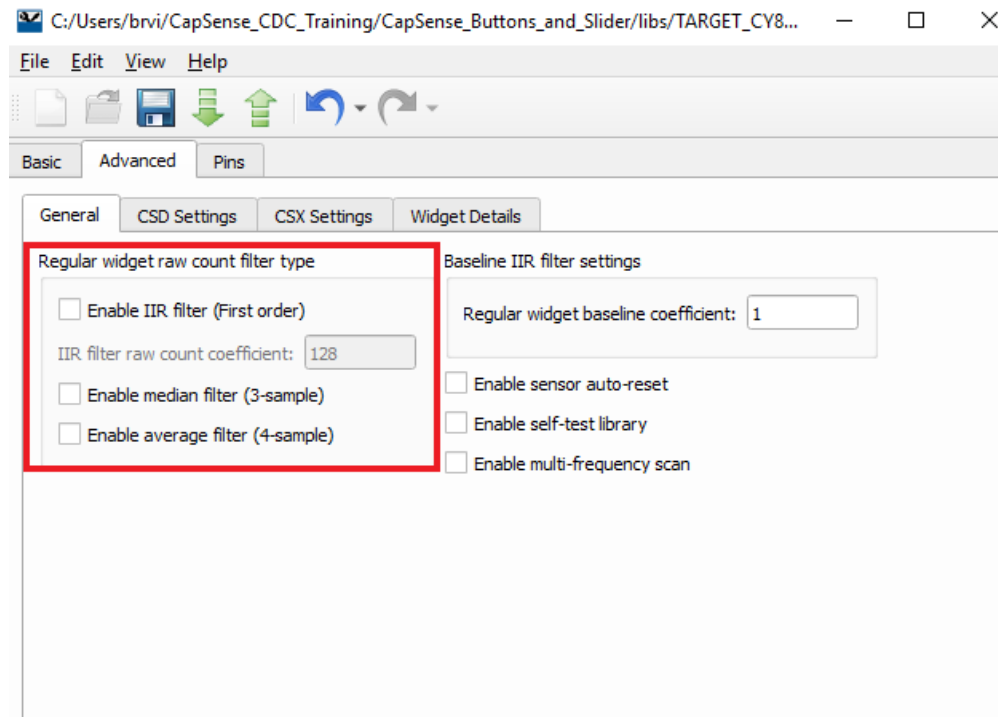


## Advanced Topics – Firmware filters

- › IIR filter - Eliminates high-frequency noise
- › Median filter - Eliminates spike noise.
- › Average filter - Eliminates periodic noise

### Limitations:

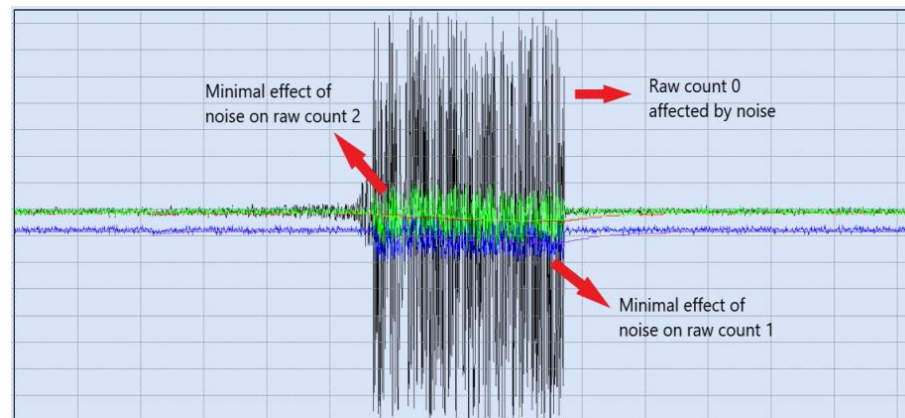
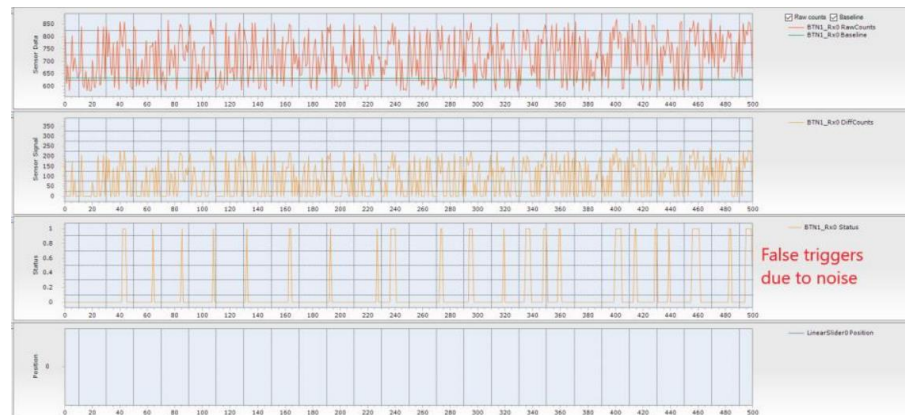
- › Increases memory consumption depending on filter
- › IIR filter slows down response
- › Increases processing time and hence power consumption





## Advanced Topics – Multi Frequency Scanning (MFS)

- › MFS is used to prevent false touch detection in the presence of external noise at a particular frequency
- › Each sensor is scanned with three different frequencies
- › Median filter is applied on the results to eliminate noise



## Advanced Topics – Multi Frequency Scanning

- › Points to consider when using MFS
  - Enabling MFS increases RAM usage by three times approximately
  - Increases the sensor scan duration by three times
  - The *SmartSense (Full Auto-Tune)* and the Multi-frequency scan features are mutually exclusive
  - Use CapSense Tuner GUI to view the rawcounts for three different channels.

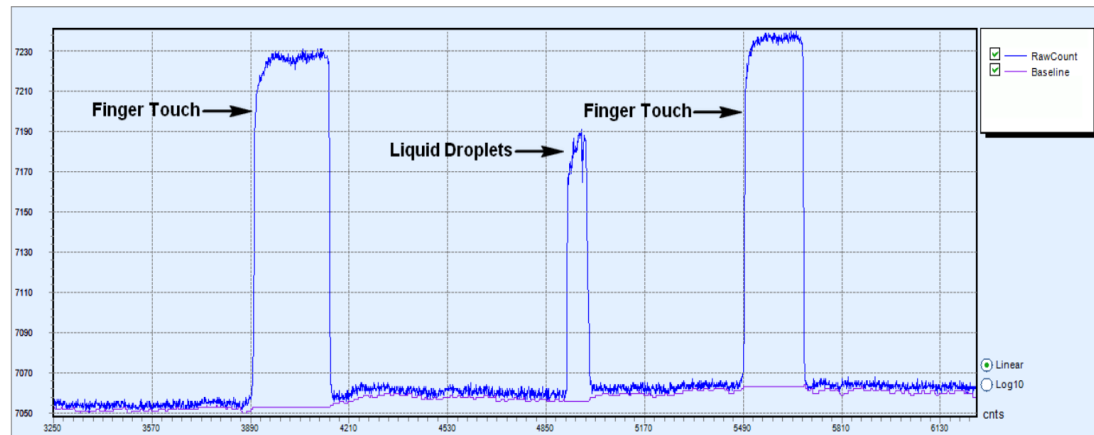
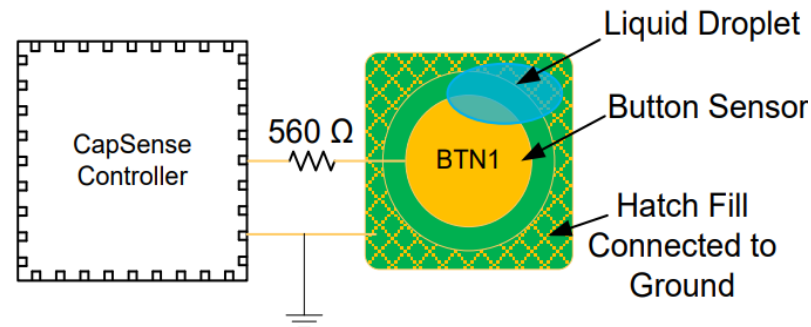
# Advanced Topics – Pipeline scanning using low level APIs

- › Custom scanning sequence
- › Reduce the total scan/ process time
- › Increase the refresh rate
- › Decrease the power consumption

```
snsIndex = 0u;
Cy_CapSense_SetupWidgetExt(CY_CAPSENSE_TOUCHPAD0_WDGT_ID, snsIndex, &cy_capsense_context); /* Trigger scanning */
Cy_CapSense_ScanExt(&cy_capsense_context);
for(;;)
{
    if (CY_CAPSENSE_NOT_BUSY == Cy_CapSense_IsBusy(&cy_capsense_context))
    {
        snsIndex++;
        if (snsIndex < cy_capsense_context.ptrWdConfig[CY_CAPSENSE_TOUCHPAD0_WDGT_ID].numSns)
        {
            /* Trigger the next sensor scanning and process previous sensor */
            Cy_CapSense_SetupWidgetExt(CY_CAPSENSE_TOUCHPAD0_WDGT_ID, snsIndex, &cy_capsense_context);
            Cy_CapSense_ScanExt(&cy_capsense_context);
            Cy_CapSense_ProcessSensorExt(CY_CAPSENSE_TOUCHPAD0_WDGT_ID, snsIndex, \
                                        CY_CAPSENSE_PROCESS_ALL, &cy_capsense_context);
        }
        else
        {
            /* The last sensor is scanned already */
            Cy_CapSense_ProcessSensorExt(CY_CAPSENSE_TOUCHPAD0_WDGT_ID, snsIndex, \
                                        CY_CAPSENSE_PROCESS_ALL, &cy_capsense_context);
            /* All sensors processed, therefore process only widget-related task */
            Cy_CapSense_ProcessWidgetExt(CY_CAPSENSE_TOUCHPAD0_WDGT_ID, CY_CAPSENSE_PROCESS_STATUS, \
                                        &cy_capsense_context);
            /* Reset sensor index to start from the first sensor */
            snsIndex = 0u;
        }
    }
}
```

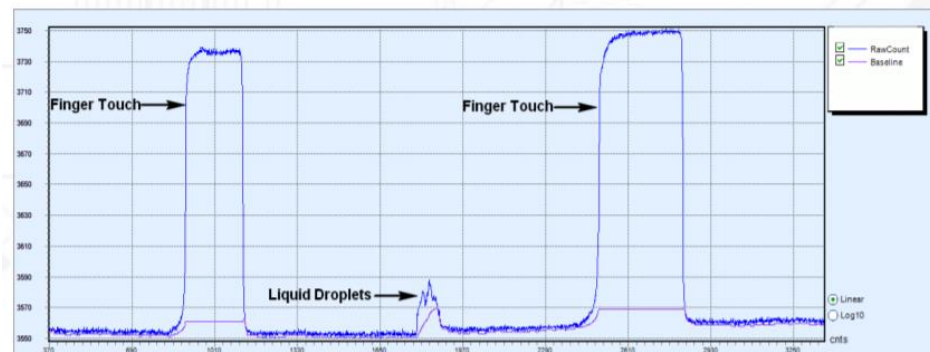
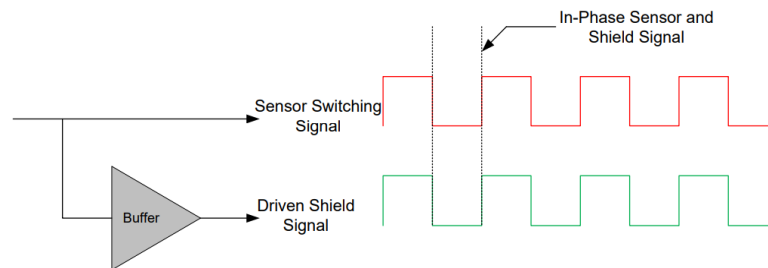
# Advanced topics - Liquid tolerance in CapSense designs

- When hatch fill around the sensor is grounded, the increase in raw count when a liquid droplet falls on the sensor surface may be equal to the increase in raw count due to a finger touch. In such a situation, sensor false triggers might occur.



# Liquid tolerance in CapSense designs - Shield Electrode

- › The driven-shield signal is a buffered version of the sensor-switching signal
- › It has the same amplitude, frequency, and phase as that of sensor switching signal
- › Buffer provides sufficient current for the driven-shield signal to drive the high parasitic capacitance of the hatch fill
- › The increase in raw count when a water droplet falls on the sensor will be very small.



Effect of Liquid Droplet when the Hatch Fill around the Sensor is Connected to the Driven-Shield

## Advanced topics – Built in Self Test (BIST)

- › Available from CapSense MW 2.10 (*to be released soon...*)
- › HW Tests – To confirm the CSD block and sensor hardware (external to chip) function correctly:
  - Chip analog-routing verification
  - Pin faults checking
  - PCB-trace opens / shorts checking
  - Sensors capacitance measurement
  - VDDA measurement.
- › FW Tests – To confirm the integrity of data used for decision-making on the sensor status
  - Global and widget specific configuration verification
  - Sensor baseline duplication
  - Sensor raw count and baseline are in the specified range.

## Advanced Topics - Gestures

- › Supports one-finger and two-finger gestures on Sliders and touchpad.

Widget Type	Gesture Groups						
	Click	One-finger Scroll	Two-finger Scroll	One-finger Flick	One-finger Edge Swipe	Two-finger Zoom	One-finger Rotate
Button							
Linear Slider	✓	✓		✓			
Radial Slider	✓						
Matrix Buttons							
Touchpad	✓	✓	✓	✓	✓	✓	✓
Proximity							

## Advanced Topics - Gestures

- › Use the high level function, `Cy_CapSense_DecodeWidget Gestures()` to process all gestures for a specified widget.
- › Gesture detection functionality requires a timestamp for its operation. The timestamp should be initialized and maintained in the application program prior to calling this function.
- › Call `Cy_CapSense_IncrementGestureTimestamp()` periodically from the application layer or register a periodic callback such as `SysTick` to keep the timestamp updated.



## Resources

---

- › [PSoC® 4 and PSoC 6 MCU CapSense® Design Guide](#)
- › [Cypress CapSense Middleware Library](#)
- › [ModusToolbox® Software Environment](#)
- › [ModusToolbox™ CapSense® Tuner Guide](#)
- › [ModusToolbox™ CapSense® Configurator Guide](#)
- › [Code Examples: Github repository](#)



Part of your life. Part of tomorrow.